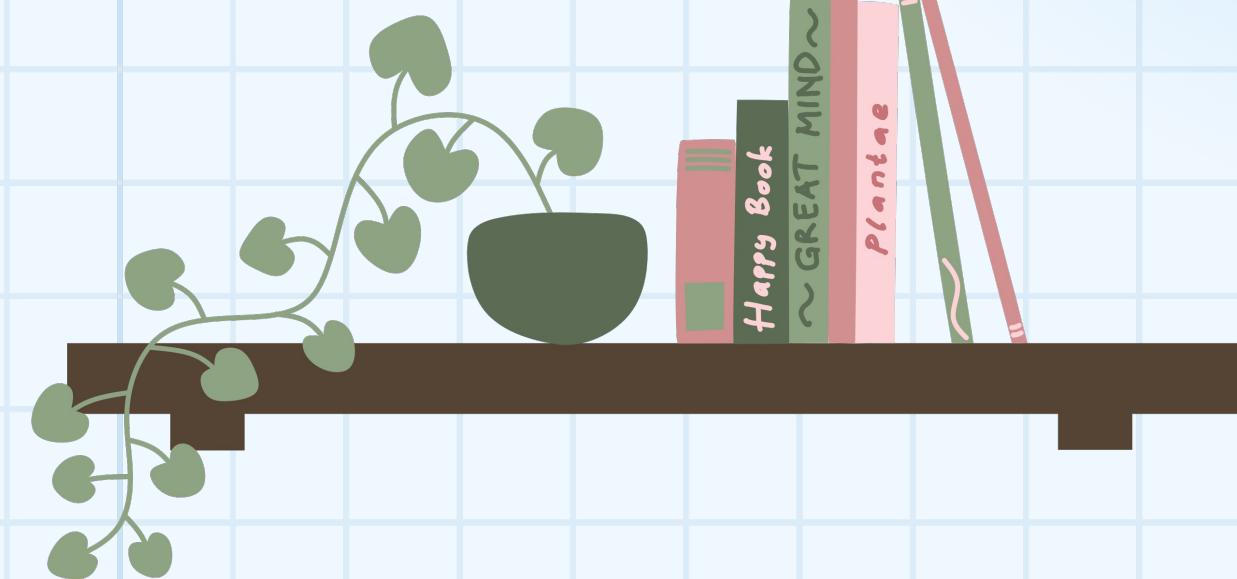
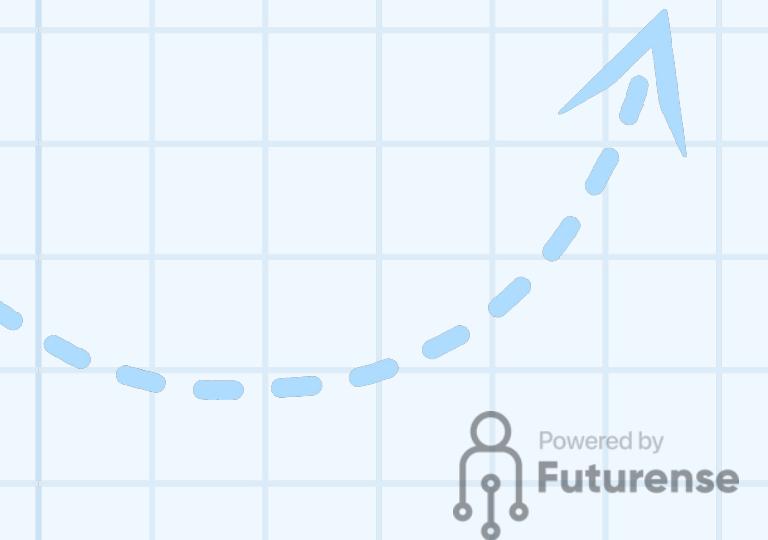
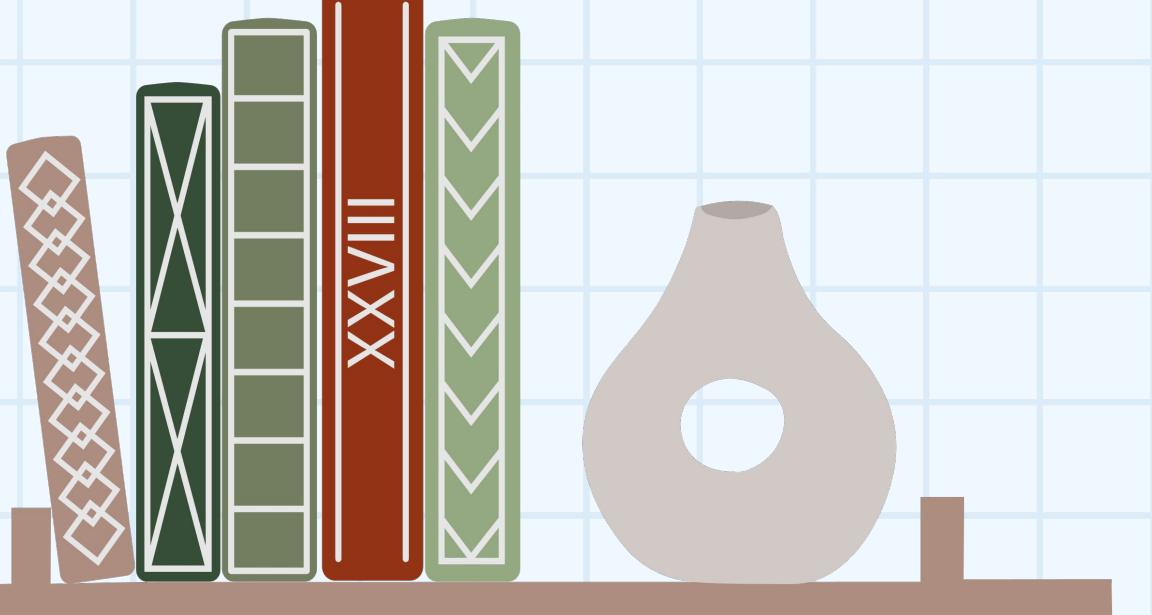


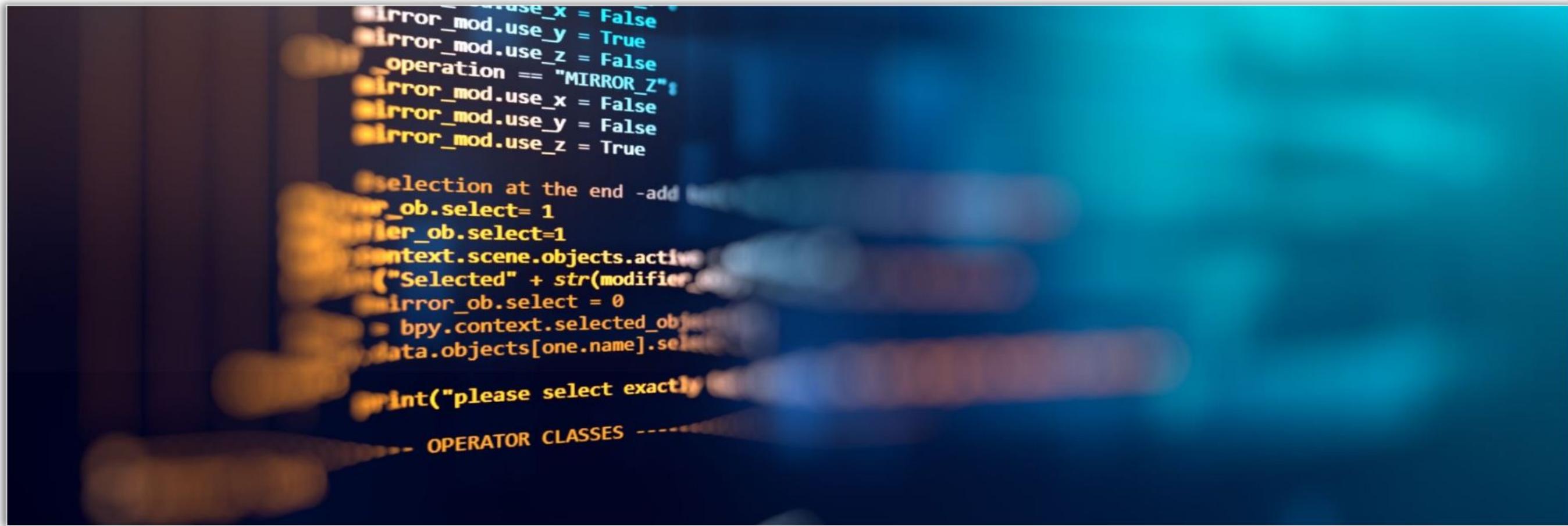


BS./BSC.IN

Applied AI and Data Science

Algorithmic Thinking & its Applications





Why learn to program?

Why learn to program?

(subtly distinct from, although a core part of, CS & IS)

Computing is worth teaching less for the subject matter itself and more for the habits of mind that studying it encourages.

“Teach computing, not Word”, the Economist
http://www.economist.com/blogs/babbage/2010/08/computing_schools

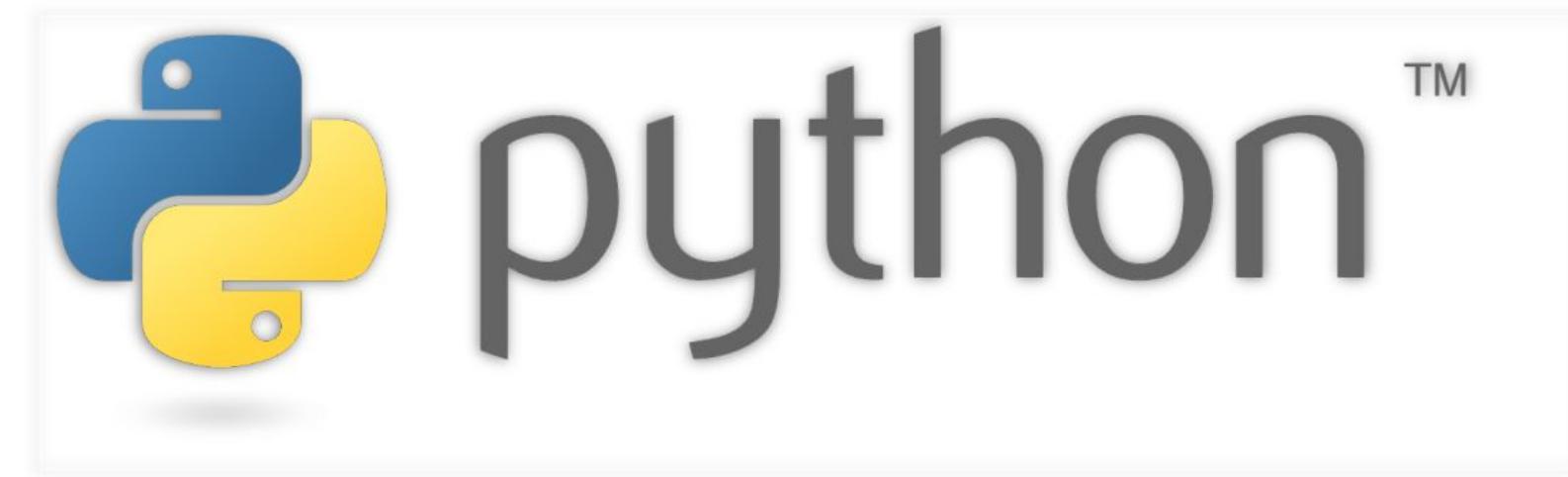
Why learn to program (continued)

[T]he seductive intellectual core of... programming: here is a magic black box. [T]ell it to do whatever you want, within a certain set of rules, and it will do it;

... within the confines of the box you are more or less God, your powers limited only by your imagination.

But the price of that power is strict discipline: you have to *really know* what you want, and you have to be able to express it clearly in a formal, structured way

that leaves no room for the fuzzy thinking and ambiguity found everywhere else in life...



Getting started with Python

Why Python?

Low overhead

- Little to learn before you start “doing”
- Easier for beginners
- Designed with “rapid prototyping” in mind

Highly relevant to non-CS majors

- NumPy and SciPy heavily used by scientists

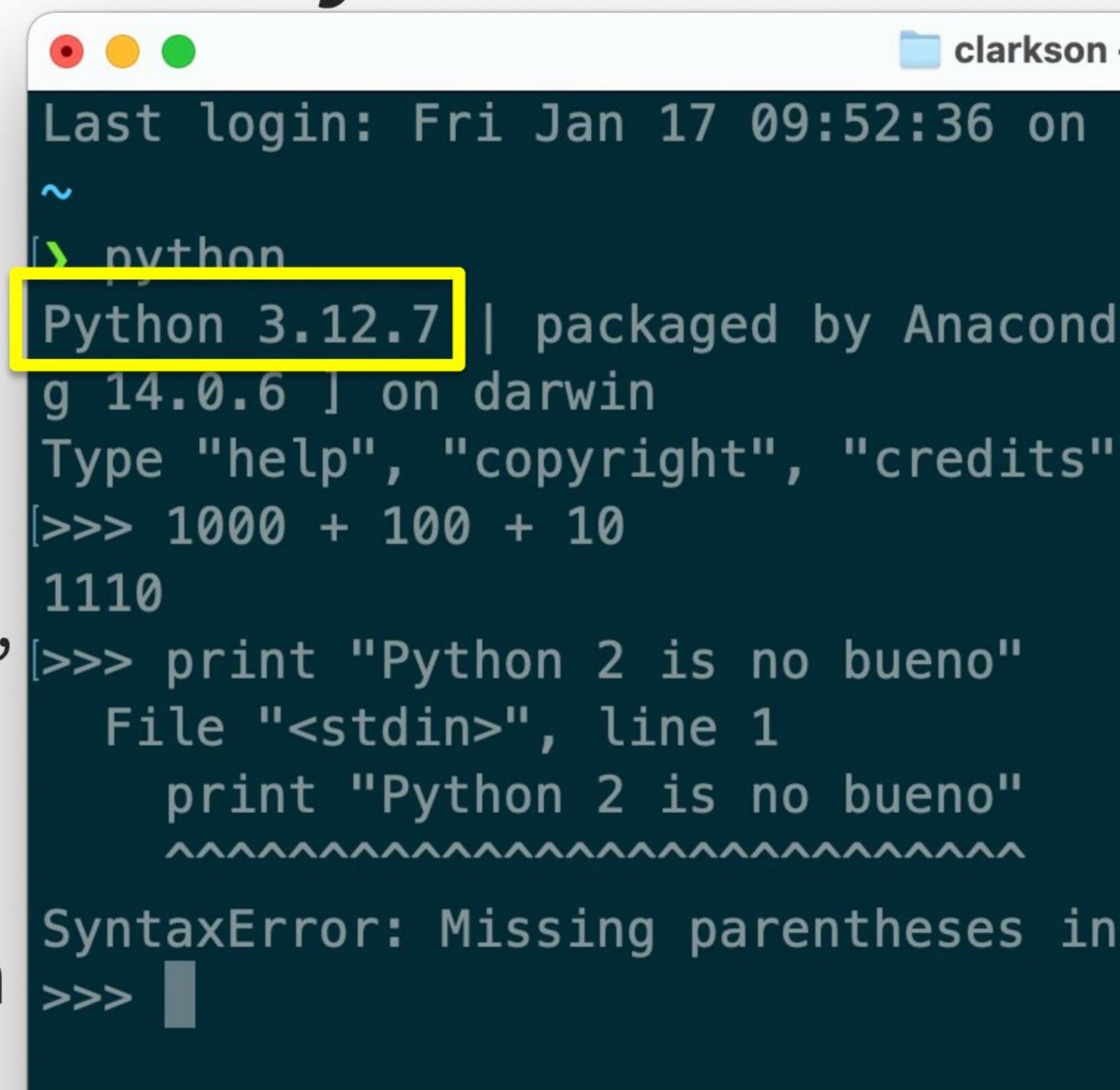
A modern language

- Popular for web applications (e.g. Facebook apps)
- Applicable to mobile app development

Getting Started with Python

- Designed to be used from the “command line”
 - OS X/Linux: **Terminal**
 - Windows: **PowerShell**
 - Preferred over **cmd**
 - Purpose of the first lab
- Install, then type “python”
 - Starts the *interactive mode*
 - Type commands at >>>
- Quit by typing `quit()` then pressing Return

```
>>> terminal time >>>
```



```
Last login: Fri Jan 17 09:52:36 on  
~  
[> python  
Python 3.12.7 | packaged by Anaconda  
g 14.0.6 ] on darwin  
Type "help", "copyright", "credits"  
[>>> 1000 + 100 + 10  
1110  
[>>> print "Python 2 is no bueno"  
  File "<stdin>", line 1  
    print "Python 2 is no bueno"  
    ^^^^^^^^^^^^^^^^^^^^^  
SyntaxError: Missing parentheses in  
>>> |
```

This class uses **Python 3**

- Make sure you are, too!

Expressions

An expression **represents** something

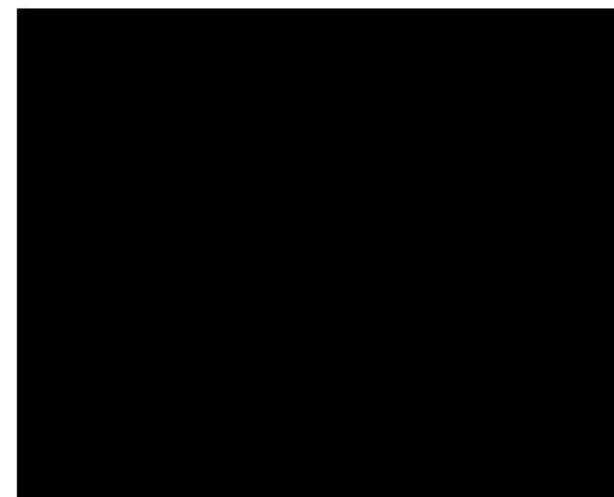
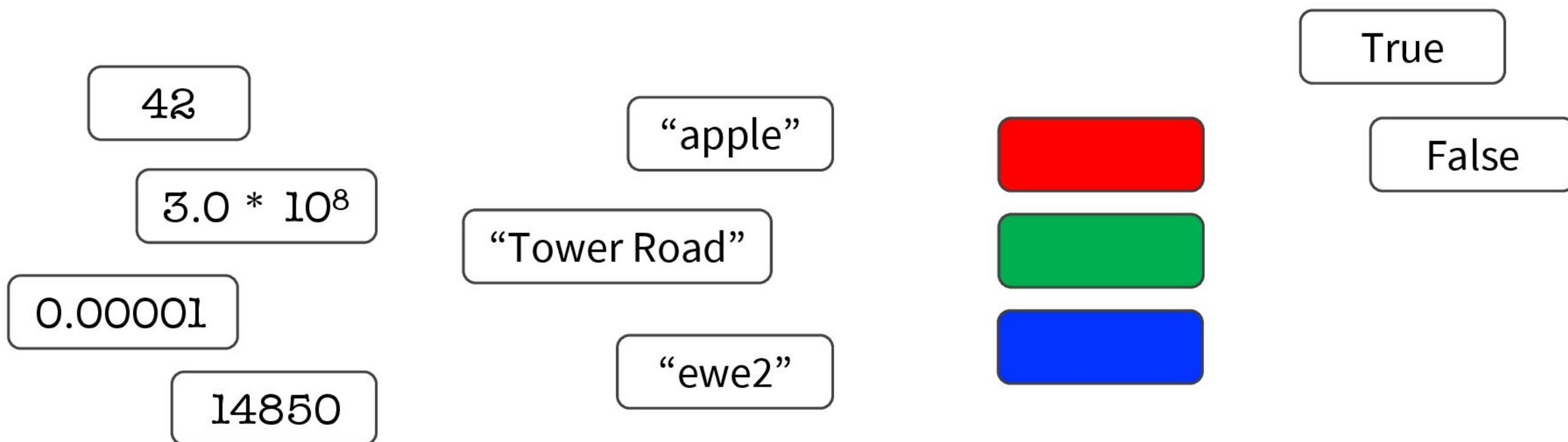
- Python ***evaluates it*** (turns it into a value)
- Similar to a calculator

Examples:

- 2.3
- $(3 * 7 + 2) * 0.1$

Storing and Computing Data

What data might we want to work with?
(What's on your computer?)



Types

A set of values & operations on these values

- Examples of operations: +, -
- Meaning of operations depends on type

Memorize this definition!

How to tell the Type of a Value

Command: type(*value*)

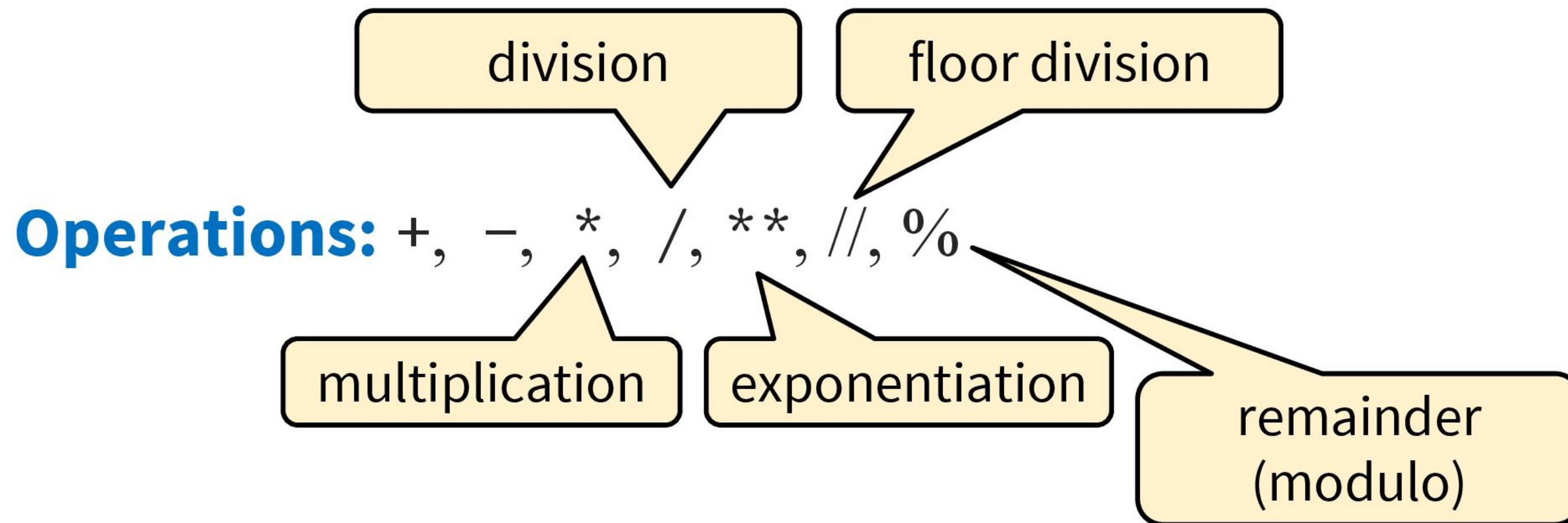
Example:

```
>>> type(2.0)
<class 'float'>
```

Type: **float** (floating point)

Values: (approximations of) real numbers

- With a “.”: a **float** (e.g., `2.0`)
- [Without a decimal: an **int** (e.g., `2`)]



Floating Point Errors

Python cannot store most real numbers exactly

- Similar to problem of writing $1/3$ with decimals

Approximation results in **representation error**

- When combined in expressions, error can get worse
- **Example:** $0.1 + 0.2$

```
>>> terminal time >>>
```

Type: **int** (integers)

Values: ..., -3, -2, -1, 0, 1, 2, 3, 4, 5, ...

More Examples: 1, 45, 43028030

(no commas or periods)

Operations: +, -, *, **, /, //, %

division (produces a float)

floor division
(produces an int)

multiplication

exponentiation

remainder
(modulo)

Note: operator meaning can change from type to type

>>> terminal time >>>

Type: **bool** (boolean)

Values: **True, False** (must be capitalized)

Often come from comparing **int** or **float** values

- Order comparison: $i < j$ $i \leq j$ $i \geq j$ $i > j$
- Equality, inequality: $i == j$ $i != j$



"=" means something else!

Operations: (comparisons above), not, and, or

- not b: **True** if b is false and **False** if b is true
- b and c: **True** if both b and c are true; **False** otherwise
- b or c: **True** if b is true or c is true; **False** otherwise

Boolean Misconceptions

Booleans expressions *sound like* English, but subtle differences cause problems:

- In English, “A or B” often means “A or B ***but not both***”
Example: “I’ll take CS 1110 or CS 1112” (but not both)
In Python, “A or B” always means “A or B ***or both***”
- In English, “A = B and C” often means “A = B and A = C”
Example: “Ithaca is cold and snowy”
 - Means: “Ithaca is cold” and “Ithaca is snowy”
 - **Does not mean:** “Ithaca is cold” and.... “snowy”

Type: **str** (string) for text

Values: any sequence of characters

Operation(s): + (catenation, or concatenation)

Again: operator + changes from type to type

String: sequence of characters in quotes

- Double quotes: " abcex3\$g<&" or "Hello World!"
- Single quotes: 'Hello World!'

Concatenation applies only to strings

- "ab" + "cd" evaluates to "abcd"
- "ab" + 2 produces an **error**



Thank you

